Those projects will do the following:

A simple page with firstname, last Name, employeeId, Age and email will be transferred to server from:

1. Asp WebForm ->WCF on IIS + angular
2. Asp WebForm ->WCF on IIS
3. ASP.Net MVC -> WCF IIS + Angular
4. ASP.Net MVC -> WCF IIS + Razor
5. ASP WebForm -> WebService (SOAP) via Ajax

The server will save data both to XML and to SQL Server

The API will support:

GetAllEmployees

AddNewEmployee ( will check if the if of th employee is exist already)

DeleteEmployee,

UpdateEmployee,

ShowEmployeeById

**Host WCF Service on IIS.**

The missing link in the exam was to host WCF under IIS to avoid having WCF running on a different port

And therefor, to avoid the need to support CORS.

Using Visual studio 2013 ( in visual studio 2015 the project build different web config, which need to be learned how to modify)

To get it work in IIS

<http://localhost/IISWCFServer/Service1.svc>

I needed to do the following:

I got the error:

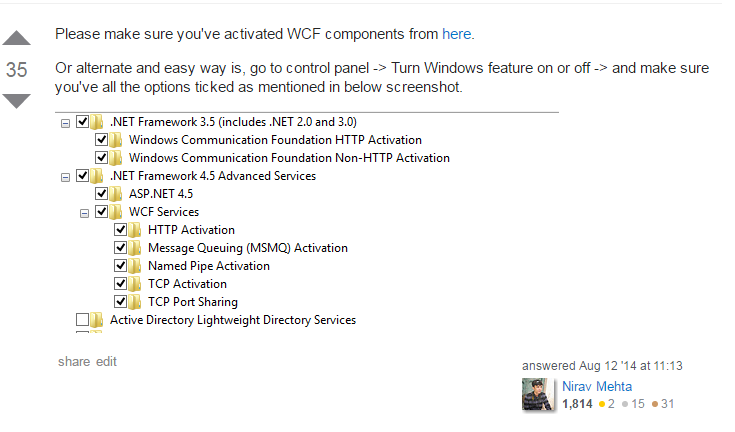
WCF – Error 500.21 Handler “svc-Integrated” has a bad module “ManagedPipelineHandler” in its module list”

1. Open Visual Studio Command Prompt
2. Type the following:
   1. *aspnet\_regiis.exe -i*

<https://blogs.msdn.microsoft.com/keithmg/2010/11/08/wcf-error-500-21-handler-svc-integrated-has-a-bad-module-managedpipelinehandler-in-its-module-list/>

Also got error 403 which needed to do:

<http://stackoverflow.com/questions/3274525/http-error-404-3-not-found-while-browsing-wcf-service-on-windows-server-2008>



Now the service is hosted in IIS as application under Default Web Site ( port 80)

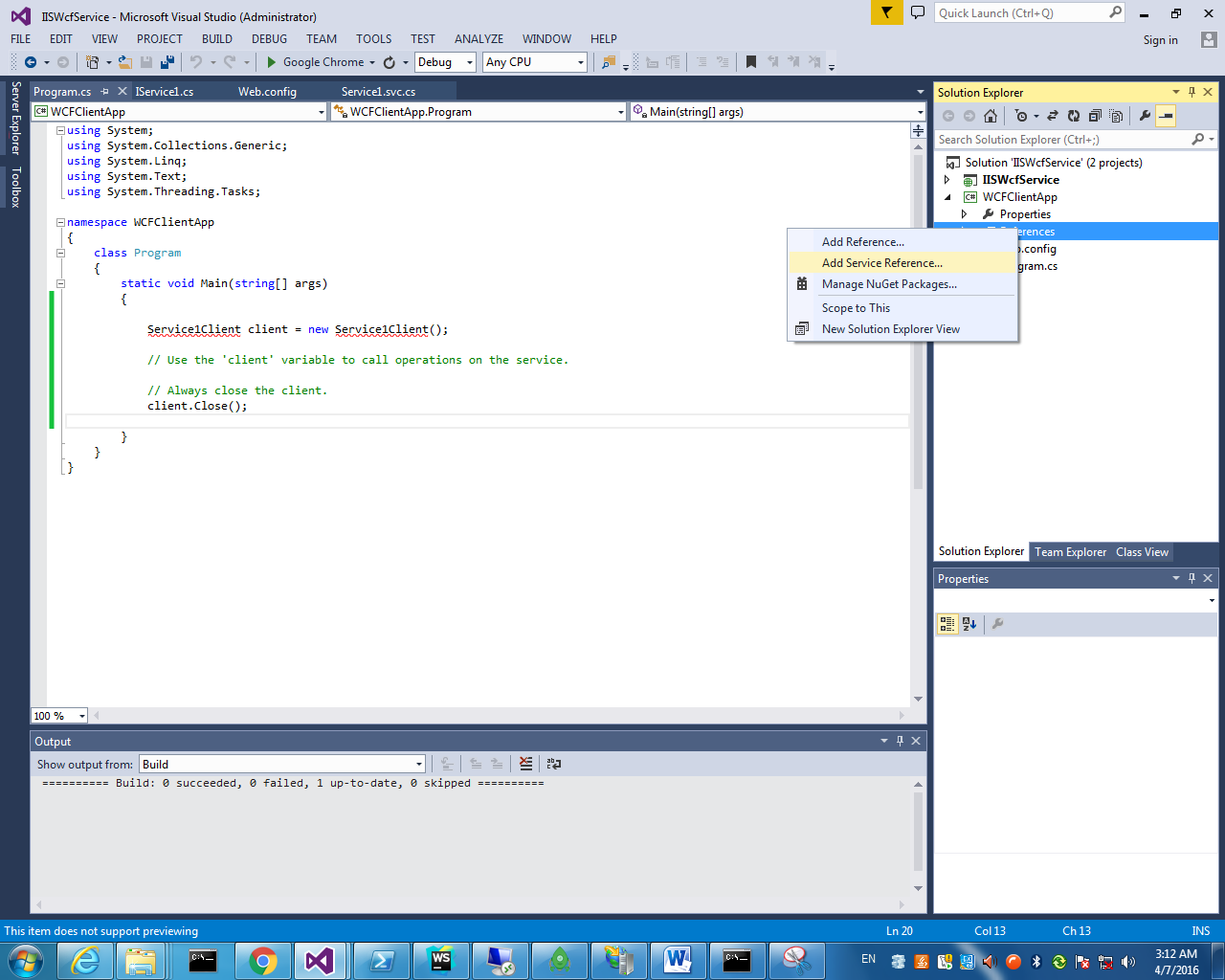


Create simple c# console application to consume the WCF service.

Later on I will build a MVC and ASP.Net Web Form application that uses this WCV service on IIS.

In the console application , right click and select

Add service reference:



Service1Client client = new Service1Client();

// Use the 'client' variable to call operations on the service.

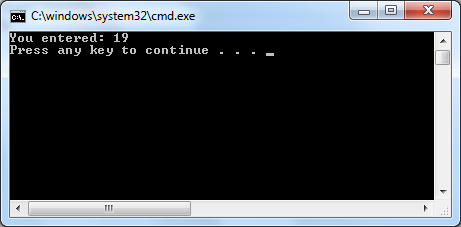
string str = client.GetData(19);

Console.WriteLine(str);

// Always close the client.

client.Close();

Result:



Now before I move on , lets add another function and check that modifying the service works.

I added:

public int Add(int a, int b)

{

return a + b;

}

Right click on the service and choose Update service reference.

There is no need to stop and start the service on IIS.

In the client app call to Add:

int r = client.Add(1, 2);

Console.WriteLine(r);

One note:

I use the IIS service reference , that means that in order to debug the WCF service itself I need to to attach to process the w3wp.exe process.

if I will have to debug the WCF code , the way is to open a new visual studio instance and to add a service reference to :

<http://localhost:57456/Service1.svc>

And leave the debugging running. ( should try later)

**Using ASP.Net MVC to work with the WCF service.**

1.Create a new ASP.Net WebForm project

2.Add bootstrap and angular

3.design the UI , left block will be add new employee , right side will be all the employees.

4. Use http to send a request to server and then call to the WCF service.

5. Use Linq to XML to save the employees.

There are two options to communicate with the WCF service:

1. Use Ajax or HTTP to the web server and from there using the same way I did in the console Application , communicate with the service.

This approach will not have the CORS issue that we will see in when it happened.

1. Use Ajax directly from the html page, or http.

<script type="text/javascript">

$(function () {

$.ajax({

type: "POST",

data:"1",

contentType: "application/json; charset=utf-8",

url: 'http://localhost/Service1.svc/GetData',

success: function (data) {

alert(data);

},

error: function (result) {

alert(result);

}

});

});

</script>

This option will cause a CORS issue if the WebForm site is still hosted on the IIS express ( running under visual studio) and the WCF service is on port 80 or different port.

The way to overcome the CORS issue is both to run on port 80 , meaning, to run directly on IIS.

But in this case we are losing the capability of working with a debugger.

The code is not in production and we want to use the debugger.

So , I don’t see any option rather than overcome the preflight issue (CORS) by adding support to the WCF service.

When adding the CORS we can use debugger for both WCF Service and the ASP.Net WebForm or any host that will consume the WCF Service.

I found the article and source code for enabling cors in WCF service:

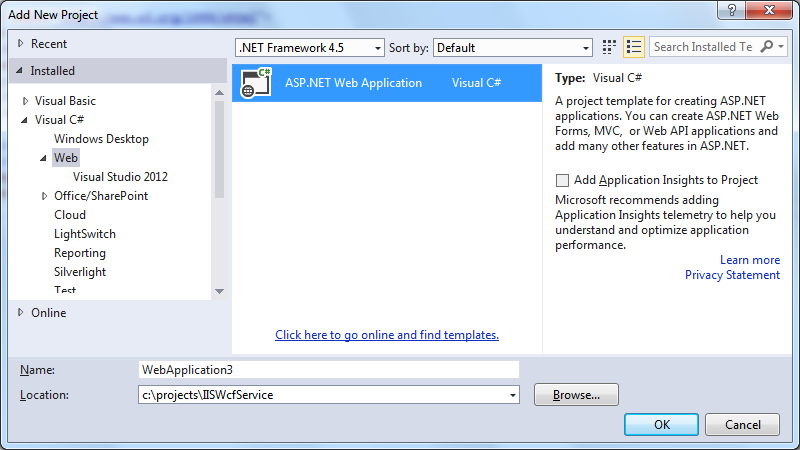
<https://code.msdn.microsoft.com/windowsdesktop/Implementing-CORS-support-c1f9cd4b>



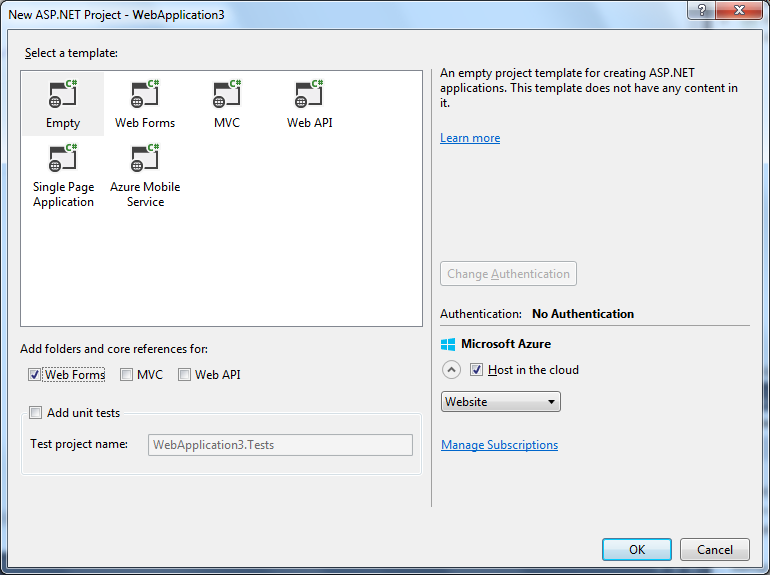
Empty WebForm + angular to consume ASMX WebService

In visual studio 2013:

Create a new empty asp.net web form project



Select Empty and check the Web Forms.



When ready add a new Web Form , lets name it index.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Index.aspx.cs" Inherits="WebApplication2.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

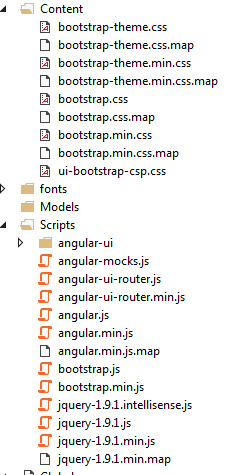
</div>

</form>

</body>

</html>

Using Nu-get package manager , add bootstrap and angularjs 1.5

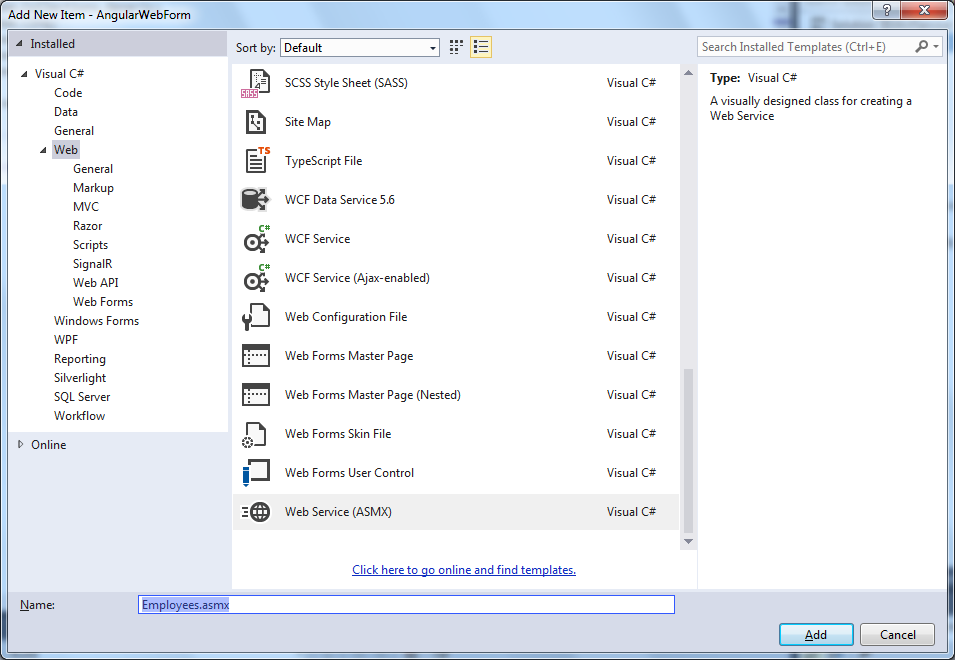


With the help of judvenkat tutorials

Consuming ASP NET Web Service in AngularJS using $http

<https://www.youtube.com/watch?v=uXbu82N0fQU&list=PL6n9fhu94yhWKHkcL7RJmmXyxkuFB3KSl&index=17>

Add a WebService (ASMX)



We want to call the web service from angular http request , so we need to enable it in the service:

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

[System.Web.Script.Services.ScriptService]

Add to index.aspx in all the script and styles of our libraries:

<title>easwdev@gmail.com - Eli Arad</title>

<meta name="viewport" content="width=device-width, initial-scale=1"/>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

<script type="text/javascript" src="Scripts/jquery-2.2.2.min.js"></script>

<link rel="stylesheet" type="text/css" href="Content/bootstrap.min.css"/>

<script type="text/javascript" src="Scripts/bootstrap.min.js"></script>

<script type="text/javascript" src="Scripts/angular.min.js"></script>

<script type="text/javascript" src="Scripts/angular-ui-router.min.js"></script>

<script type="text/javascript" src="Scripts/angular-cookies.min.js"></script>

<script type="text/javascript" src="Client/app.js"></script>

<script type="text/javascript" src="Client/routing.js"></script>

<script type="text/javascript" src="Client/Controllers/homeController.js"></script>

The basic code for bootstrap angular is the following:

App.Js:

'use strict';

var app = angular

.module('app', [

'ui.router',

'ngCookies'

]);

Routing.js:

app.config(['$stateProvider', '$urlRouterProvider', '$httpProvider',

function ($stateProvider, $urlRouterProvider, $httpProvider) {

$urlRouterProvider.otherwise('/');

$stateProvider.state('/', {

url: '/',

templateUrl: '/Client/Views/home.html'

});

}]);

The home.html page:

<div class="container-fluid">

<div class="row">

<div class="col-md-4">

This is home from angular

</div>

<div class="col-md-4">

</div>

</div>

</div>

Modify index.aspx to the following code:

<body data-ng-app="app">

<form id="form1" runat="server">

<div ui-view="">

</div>

</form>

</body>

Two more things before we start:

To tell the web form to load first the index.aspx:

Add into webConfig the following lines:

<system.webServer>

<defaultDocument>

<files>

<add value="Index.aspx" />

</files>

</defaultDocument>

<directoryBrowse enabled="false" />

</system.webServer>

Nice to the have is the locationProvider for html5Mode to remove the # from the url.

$locationProvider.html5Mode(true);

Add it in routing.js

app.config(['$stateProvider', '$urlRouterProvider', '$httpProvider', '$locationProvider',

function ($stateProvider, $urlRouterProvider, $httpProvider, $locationProvider) {

$urlRouterProvider.otherwise('/');

$stateProvider.state('/', {

url: '/',

templateUrl: '/Client/Views/home.html'

});

$locationProvider.html5Mode(true);

}]);

Now we are ready to start with the actual application.

Mission as noted , to create a page with employee ,add and show.

I like to use as syntax for controllers:

<div class="container-fluid" ng-controller="homeController as vm">

I add angular validation

[WebMethod]

public void HelloWorld()

{

Context.Response.Write("Hello world");

}

One way to response with a json is to use the Context.Response.Write

And the return value should be void,

The default of the web method is to response with XML

[WebMethod]

public string HelloWorld()

{

return "Hello World";

}

**Another example that return a json:**

$http.get('Employees.asmx/GetPersonInfo').then(function (result) {

console.log(result.data);

}).catch(function (result) {

alert(result.data);

})

And the server side is:

[WebMethod]

[System.Web.Script.Services.ScriptMethod(UseHttpGet = true, ResponseFormat = System.Web.Script.Services.ResponseFormat.Json)]

public void GetPersonInfo()

{

// ASP.NET will automatically JSON serialize this as well.

object e = new {

FirstName = "Eli" ,

LastName = "Arad" ,

Address = "Enterprise E",

City = "Deck 9",

State = "Israel",

ZipCode = "001"

};

var oSerializer = new System.Web.Script.Serialization.JavaScriptSerializer();

Context.Response.Write(oSerializer.Serialize(e));

}

To pass complete JSON we can use the dynamic data type in c#

[WebMethod]

public void AddNewEmployee(dynamic data)

{

foreach (KeyValuePair<string, object> kvp in data)

Console.WriteLine("{0} = {1}", kvp.Key, kvp.Value);

}

And in angular , the submit form will do the following:

$http.post('Employees.asmx/AddNewEmployee', { data: vm.user }).then(function (result) {

console.log(result.data);

}).catch(function (result) {

alert(result.data);

})

Next step:

Save the data as XML

For that we have KodVenKat to help (<https://www.youtube.com/watch?v=CK2yLQU_hqA&list=PL6n9fhu94yhX-U0Ruy_4eIG8umikVmBrk&nohtml5=False>)

There are many ways to work with XML in c#, lets uses linq to xml.

Check if email is exist using linq to xml.

bool IsEmailExists(string email)

{

var c = (from student in XDocument.Load(fileName).Descendants("Employee")

where student.Element("email").Value == email

select student.Element("email").Value).Count();

Console.WriteLine(c);

return c > 0 ? true : false;

}

One note:

For post the function should return string

For get the function should return void and use the Context .Response.Write

Get All employees.

In the asmx web service we need to create a get function that will return a json of all emplyees from XML file.

I am using a generic directive for updating the view that I wrote in order to use a schema based directive.

The directive will hook , watch an event to redraw it self once the user entered a new record.

scope.$on("toggleAnimation", function (event, args) {

Get the display again

});